(12) UK Patent Application (19) GB (11) 2 290 395 (13) A

(21) Application No 9411670.4

(22) Date of Filing 10.06.1994

(71) Applicant(s)
Advanced Risc Machines Limited

(Incorporated in the United Kingdom)

Fulbourn Road, Cherry Hinton, CAMBRIDGE, CB1 4JN,
United Kingdom

(72) Inventor(s)
Edward Colles Nevill

(74) Agent and/or Address for Service
D Young & Co
21 New Fetter Lane, LONDON, EC4A 1DA,
United Kingdom

(51) INT CL6
G06F 9/30

(52) UK CL (Edition N)
G4A AFP

(56) Documents Cited
GB 2016755 A   EP 0324308 A2   EP 0306920 A2

(58) Field of Search
UK CL (Edition M) G4A AFP
INT CL5 G06F 9/30
On-line database: WPI

(54) Interoperability with multiple instruction sets

(57) Data processing apparatus comprises a processor core 10 for executing successive program instruction words of a predetermined plurality of instruction sets; a memory 20 for storing program instruction words to be executed; a program counter register 130 for indicating the address of a next program instruction word in the memory; means 140 for modifying the contents of the program counter register in response to a current program instruction word; and control means 90, 100, 110, responsive to one or more predetermined indicator bits T of the program counter register, for controlling the processor core to execute program instruction words of a current instruction set selected from the predetermined plurality of instruction sets and specified by the state of the one or more indicator bits T.
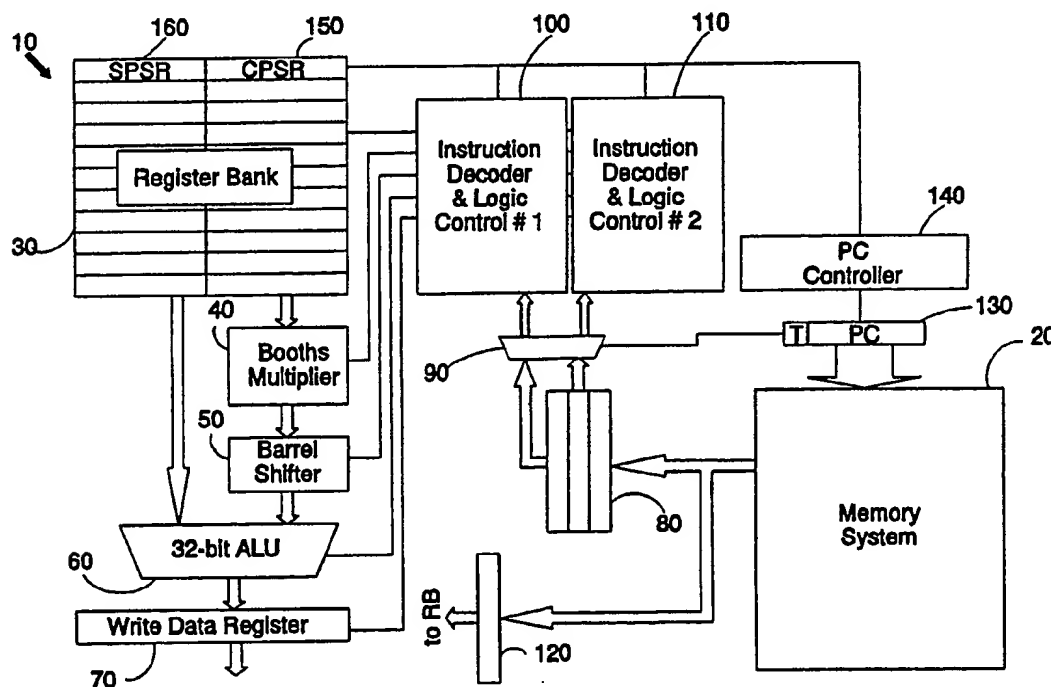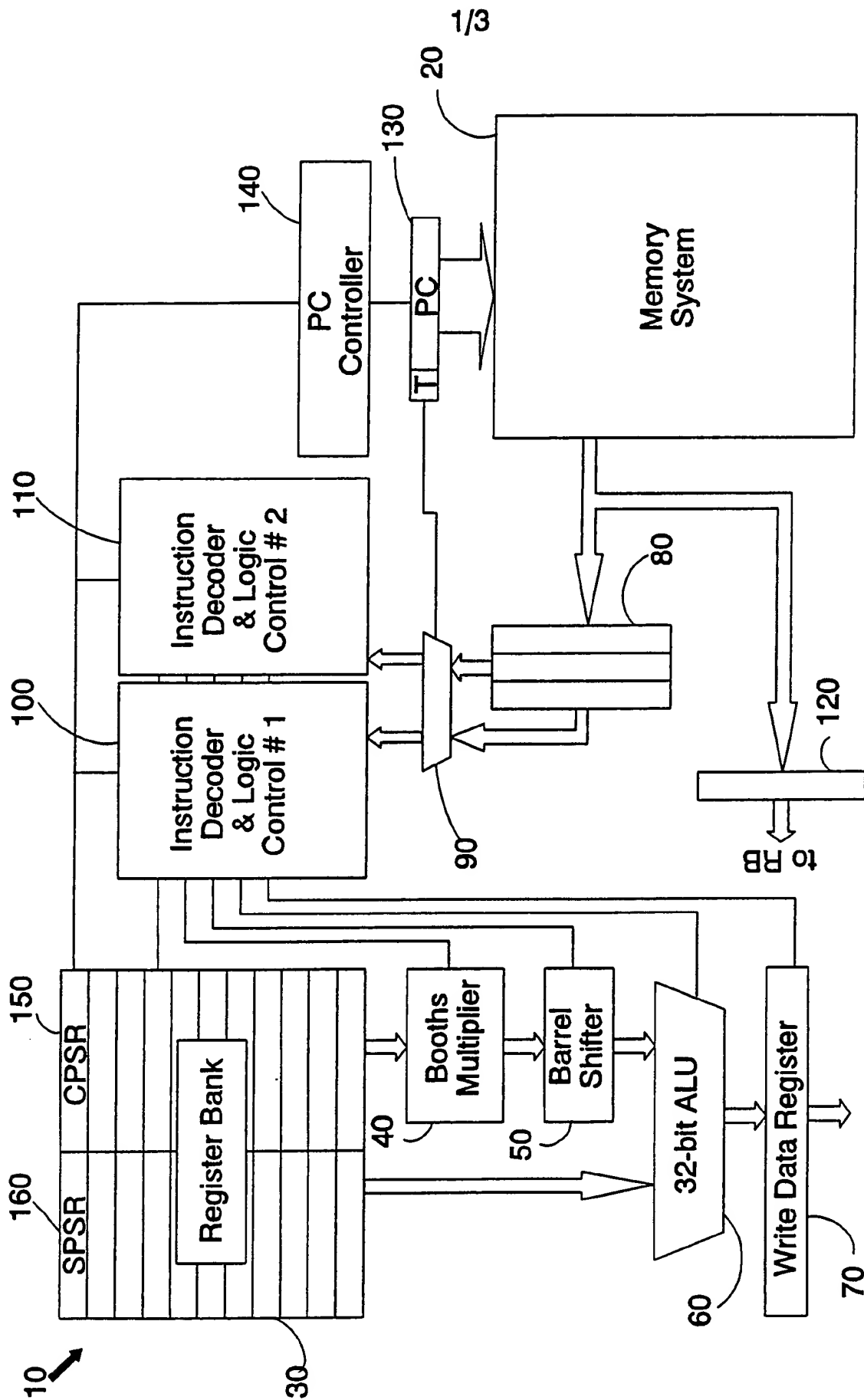
Fig.1

GB 2 290 395 A

Fig.1

130'

| 1 | 0000000000000000xxxxxxxxxxxxxx0 |

T bit   Memory address

Fig. 2

130"

| 0000000000000000xxxxxxxxxxxxxx | 1 |

Memory address   T bit

Fig. 3

32 bit
Instruction Set

16 bit
Instruction Set

Begin — 200

Data
Processing — 210

Branch to
Badd(1) + 1 — 220

Data
Processing — 230

Branch to
Badd(2) + 0 — 240

Data
Processing — 250

End — 260

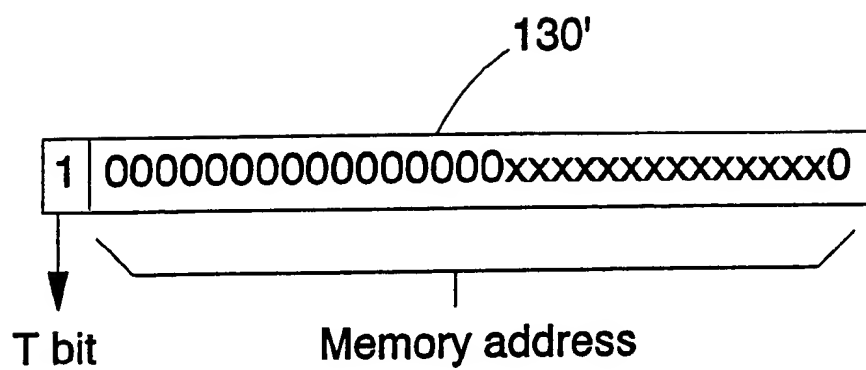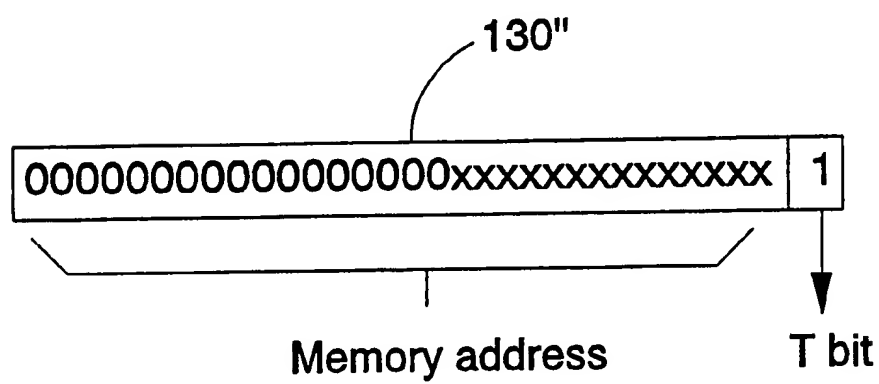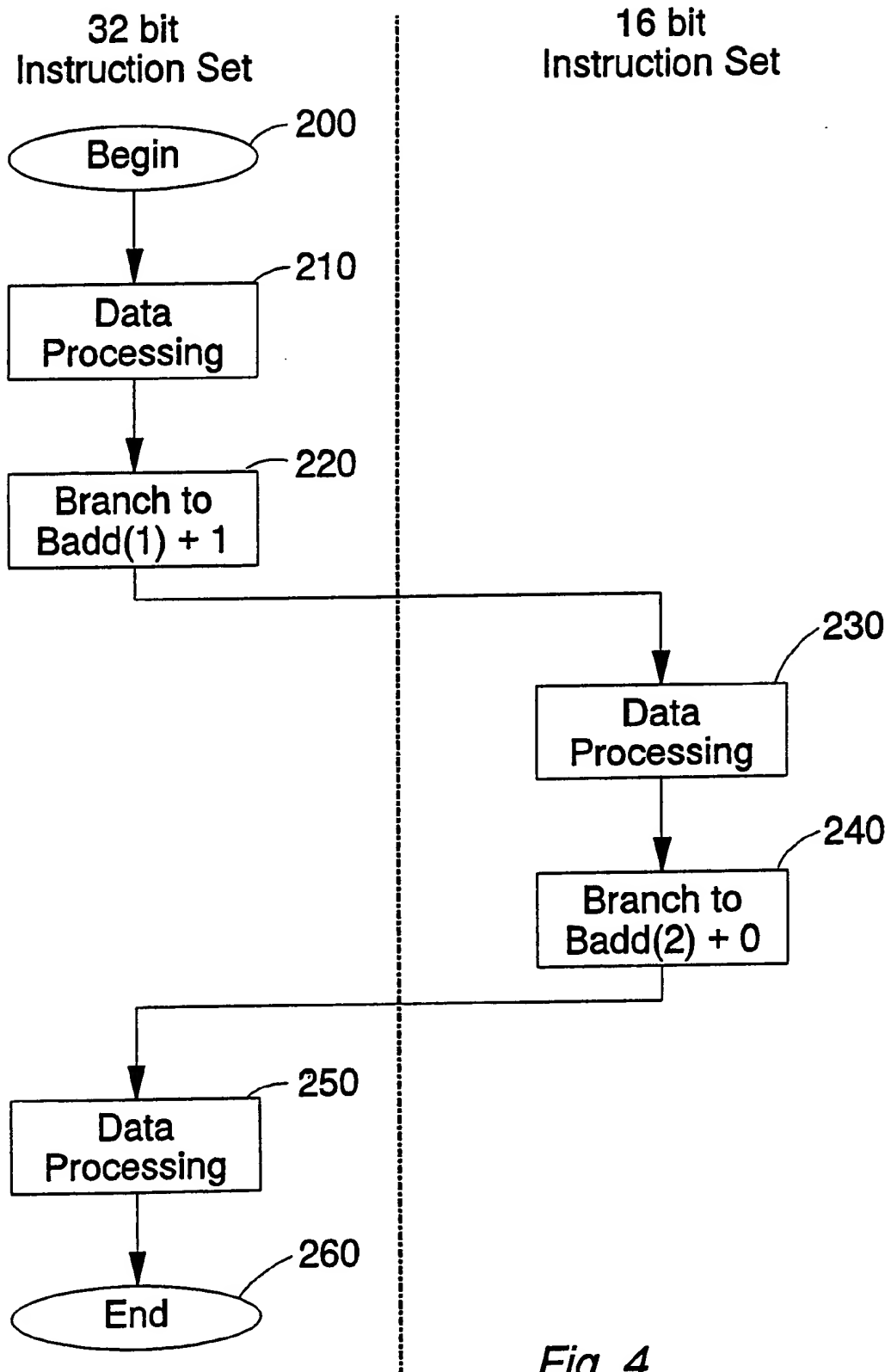*Fig. 4*

INTEROPERABILITY WITH MULTIPLE INSTRUCTION SETS

This invention relates to the field of data processing, and in particular to data processing using multiple sets of program instruction words.

Data processing systems operate with a processor core acting under control of program instruction words, which when decoded serve to generate core control signals to control the different elements in the processor to perform the necessary operations to achieve the processing specified in the program instruction word.

It is known to provide systems that execute program instruction words from two or more instruction sets, with means being provided to switch between use of the different instruction sets. The VAX11 computers of Digital Equipment Corporation have a VAX instruction mode and a compatibility mode that enables them to decode the instructions for the earlier PDP11 computers.

In order to switch between the different instruction sets, an instruction set switch may be hard-wired into the processor core necessitating a physical rewiring of the processor to switch instruction sets. Alternatively, a processor register may be used to specify the current instruction set to be used. In this case, the current instruction set can be selected by the operating software, by writing an instruction set-specifying value to that processor register. However, as described below, this technique requires additional program instruction words, which in turn require extra time during preparation of the software and extra memory space to store the program instruction words.

In order to execute a piece of code, a processor capable of using two or more instruction sets must have two pieces of information:

1)   The address of the code in memory; and

2)   The instruction set to use (i.e. the instruction set in which the code is written)

Typically, in the previously proposed processors, a call to a routine in a different instruction must be performed as described below.

1)   The subroutine call is diverted from its original destination to an automatically generated instruction set

selection sequence or veneer.

2) The veneer must then accomplish the following
    - Save the context of the caller
    - Select the correct instruction set
5   - Call the original routine
    - On return from the original routine, select the original
    instruction set
    - Restore the callers context.

This process can be made relatively transparent to the programmer
10 by use of a conventional software tool called a Linker. However, the
process has a five instruction overhead per routine which is called
from a different instruction set, and it also introduces a significant
processing overhead.

This invention provides a data processing apparatus comprising:
15   a processor core having means for executing successive program
instruction words of a predetermined plurality of instruction sets;

a data memory for storing program instruction words to be
executed;

a program counter register for indicating the address of a next
20 program instruction word in the data memory;

means for modifying the contents of the program counter register
in response to a current program instruction word; and

control means, responsive to one or more predetermined indicator
bits of the program counter register, for controlling the processor
25 core to execute program instruction words of a current instruction set
selected from the predetermined plurality of instruction sets and
specified by the state of the one or more indicator bits of the program
counter register.

With the invention, a control flag or flags to select a current
30 instruction set is provided in the program counter register. This
allows the current instruction set to be changed when a new value is
written into the program counter register, for example as part of the
execution of a branch instruction.

The invention recognises that if the required instruction set and
35 the next instruction address are encoded in separate processor
registers as in the previously proposed processors described above (an
instruction set register and a program counter register), it becomes

difficult to change between instruction sets as the two separate registers have to be updated to accomplish a call to a section of code written in a different instruction set.

As an example, consider a program which is to perform a sorting or collation function. Typically this will call a generic sort routine to perform the sort. As this sort routine is generic, it must be capable of sorting in any given sequence. For example, it may be called to sort items in numerical order, alphabetical order, case insensitive alphabetical order, or any other order specified by the programmer. The means by which the programmer specifies the sorting order is to pass the address of a routine (called a compare routine) to the sort routine. This compare routine will then be called by the sort routine and will return a value to indicate whether, given two items of data, the first should be placed before or after the second in the sorted sequence.

If just the address of the compare routine is passed to the sort routine then the sort routine has no way of knowing which instruction set should be selected when the routine is to be called. If the wrong instruction set is current when an attempt is made to execute the compare routine, the results can be dramatically unsuccessful. Extra information must be passed to the sort routine to tell it what instruction set should be in force when the compare routine is called. However, many existing programs written in high level languages such as C & C++ make assumptions that all the information necessary to uniquely identify a target routine (in this case the address and the instruction set information) can be represented in a single machine word.

The invention addresses these problems by defining a predetermined bit or bits of the program counter register (PC) to indicate the instruction set to be used. In the specific example given above, the address of the compare routine passed to the sort routine can have the required instruction set encoded in the predetermined bit or bits of that address. The address, including the indicator bit or bits, is then simply moved to the program counter register when the compare routine is called.

Although certain bits of the program counter register can be reserved for use as the indicator bits, an alternative approach is to store portions of code to be executed using the various instruction

sets in corresponding memory areas, so that while those memory areas are being accessed the program counter will contain a particular range of values specifying the appropriate instruction set to be used.

In order to decode instructions from the different instruction sets, it is preferred that the apparatus comprises a first instruction decoder for decoding program instruction words of the first instruction set; and a second instruction decoder for decoding program instruction words of the second instruction set; and that the control means is operable to control either the first instruction decoder or the second instruction decoder to decode a current program instruction word.

Preferably, program instruction words of the first instruction set are X-bit program instruction words; and program instruction words of the second instruction set are Y-bit program instruction words; where Y is different to X. In this way, a common processor core can be programmed with either an instruction set having longer program instruction words and allowing potentially more powerful and involved instructions, or an instruction set having shorter program instruction words, thus saving memory space where a potentially more limited instruction set can be tolerated.

In one preferred embodiment, the one or more bits of the program counter register are one or more most significant bits of the program counter register. In a program counter register of, say, 32 bits, the highest order bits are seldom required since the maximum memory space that can be addressed by such a large program counter register is much more than the memory space normally used.

Alternatively, in another preferred embodiment, the one or more bits of the program counter register are one or more least significant bits of the program counter register. In this case, these bits are often not used where the minimum length of program instruction words or data words is at least two bytes.

In order to avoid invalid addresses in the data memory being accessed, it is preferred that means are provided for accessing a program instruction word stored in the data memory, the accessing means not being responsive to the one or more bits of the program counter register.

An embodiment of the invention will now be described, by way of example only, with reference to the accompanying drawings, throughout

which like parts are referred to by like references, and in which:

Figure 1 is a schematic diagram of a data processing apparatus having a processor core and a memory system;

Figures 2 and 3 are schematic diagrams of program counter registers; and

Figure 4 is a schematic flow diagram illustrating transitions between two instruction sets using the program counter register of Figure 3.

Figure 1 is a schematic diagram of a data processing apparatus having a processor core 10 coupled to a memory system 20.

The processor core 10 includes a register bank 30, a Booths multiplier 40, a barrel shifter 50, a 32-bit arithmetic logic unit (ALU) 60 and a write data register 70. Between the processor core 10 and the memory system 20 are: an instruction pipeline 80, a multiplexer 90, a first instruction decoder 100, a second instruction decoder 110, and a read data register 120.

A program counter (PC) register 130, which is part of the processor core 10, is shown addressing the memory system 20. A program counter controller 140 serves to increment the program counter value within the program counter register 130 as each instruction is executed and a new instruction must be fetched for the instruction pipeline 80. Also, when a branch instruction is executed, the target address of the branch instruction is loaded into the program counter 130 by the program counter controller 140.

The processor core 10 incorporates 32-bit data pathways between the various functional units. In operation, instructions within the instruction pipeline 80 are decoded by either the first instruction decoder 100 or the second instruction decoder 110 (under the control of the multiplexer 90) to produce various core control signals that are passed to the different functional elements of the processor core 10. In response to these core control signals, the different portions of the processor core conduct 32-bit processing operations, such as 32-bit multiplication, 32-bit addition and 32-bit logical operations.

The register bank 30 includes a current programming status register (CPSR) 150 and a saved programming status register (SPSR) 160. The current programming status register 150 holds various condition and status flags for the processor core 10. These flags may include

processing mode flags (e.g. system mode, user mode, memory abort mode, etc.) as well as flags indicating the occurrence of zero results in arithmetic operations, carries and the like. The saved programming status register 160 (which may be one of a banked plurality of such saved programming status registers) is used to store temporarily the contents of the current programming status register 150 if an exception occurs that triggers a processing mode switch.

The program counter register 130 includes an instruction set flag, T. This instruction set flag is used to control the operation of the multiplexer 90, and therefore to control whether the first instruction decoder 100 or the second instruction decoder 110 is used to decode a current data processing instruction. In the present embodiment, two instruction sets are used: a first instruction set comprises 32-bit program instruction words and is decoded by the first instruction decoder 100, and a second instruction set comprises 16-bit program instruction words and is decoded by the second instruction decoder 110. The core control signals generated by the first instruction decoder 100 and the second instruction decoder 110 are compatible with the various functional units of the core 10.

The use of two instruction sets of different program instruction word length allows a common processing core 10 to be programmed with either the first instruction set having longer words and allowing potentially more powerful and involved instructions, or the second instruction set having shorter program instruction words, thus saving memory space where a potentially more limited instruction set can be tolerated.

The provision of an instruction set flag T enables the second instruction set to be non-orthogonal to the first instruction set. This is particularly useful in circumstances where the first instruction set is an existing instruction set without any free bits that could be used to enable an orthogonal further instruction set to be detected and decoded.

The instruction set flag T is "hidden" in normally unused bits of the program counter register. This means that the T flag can be set or reset by the program counter controller 140, but the state of the T flag need have no direct effect on the operation of the memory system 20 and the instruction pipeline 80.

Figures 2 and 3 are schematic diagrams of program counter registers illustrating two possible methods in which the T bit can be encoded into the program counter register. These two methods involve encoding the T bit either as a normally unused high order (most significant) bit of the program counter register or as a normally unused low order (least significant) bit of the program counter register.

Figure 2 is a schematic diagram of a program counter register 130' in which the T bit is encoded as the highest order bit of the program counter register.

The program counter register is a 32-bit register, which allows 2^{32} bytes to be addressed in the memory system 20. However, since this equates to 4 gigabytes of addressable memory space, it is extremely unlikely that the full address range made possible by the 21-bit program counter register will be required.

Accordingly, the T bit in Figure 2 is encoded as the highest order bit of the program counter register 130'. This still allows 2 gigabytes of memory to be addressed, although in practice much less than this amount of memory will normally be addressed, and other high order bits of the program counter register may well be zeros (as shown in Figure 2).

A problem which must be overcome is that when the T bit is set, the program counter register 130' may well point to a memory address which is far in excess of the address range of the memory system 20. In other words, the memory address pointed to by the 32-bits of the program counter register 130 is an invalid address as far as the memory system 20 is concerned.

This problem can be overcome in two straightforward ways. In one technique, the highest order bit (the T bit) of the program counter register 130' is simply not supplied as an address bit to the memory system 20. Alternatively, the address decoding within the memory system 20 may detect only a certain number of lowest order bits (e.g. the lowest order 24 bits to address a 16 megabyte address space), with the state of the remaining higher order bits being irrelevant to the decoded address. This is a standard technique in memory address decoding when it is known in advance that only a certain number of address bits will be required.

As described above, the T bit is passed from the program counter register 130' to the multiplexer 90, and determines the routing of instructions to either the first instruction decoder 100 or the second instruction decoder 110.

5   Figure 3 is a schematic diagram of a second program counter register 130'', in which the instruction set switching T bit is encoded as the lowest order bit of the program counter register.

The lowest order bit of the program counter register is normally unused in a processor in which the minimum instruction or data word 10 size is at least two bytes (16 bits in this case). Accordingly, in the present embodiment the instruction program words may be either 32 bits long (4 bytes) or 16 bits long (2 bytes) so the addresses supplied from the program counter 130 to the memory system 20 will always be a multiple of two and will therefore have a zero as the least significant 15 bit of the address.

The least significant bit of the program counter register 130'' is used to store the T bit, which is supplied to the multiplexer 90 as described above. Also as described above, the lowest order bit of the program counter register 130'' is not supplied to the memory system, in 20 order that invalid addresses are not accessed by the memory system 20.

The fact that the program counter 130 is controlled by the program counter controller 140 means that the T bit can be set as part of a branch instruction carried out by the core 10. For example, if the T bit is currently set to indicate the use of the first (32-bit) 25 instruction set and it is desired to branch to a portion of a code employing the second (16-bit) instruction set, then a branch instruction can be executed to jump to the address of the 16-bit code to be executed and simultaneously to change the T bit in the program counter register. In particular, in the arrangement shown in Figure 2 30 in which the T bit is encoded as the highest order bit of the program counter register 130', a branch instruction could set the T bit to 1 by branching to (target address plus 10000000000000000000000000000000). Alternatively, in order to set the T bit to 1 in the program counter register 130'' of Figure 3, a branch instruction could take the form of 35 branch to (target address plus 1). A similar arrangement could be used to change the T bit back to a zero.

This process is illustrated schematically in Figure 4, which is

a flow diagram illustrating transitions between the 32-bit instruction set and the 16-bit instruction set using the program counter register 130'' of Figure 3. In Figure 4, when the T bit is set to 1, this signifies that the 16-bit instruction set is to be used.

Referring to Figure 4, the processing begins 200 in the 32-bit instruction set. After various data processing operations 210, a branch instruction 220 is executed to branch to an address Badd(1)+1. The address Badd(1) is the start address of a portion of code using the 16-bit instruction set, and the extra "+1" is used to switch the T bit to indicate that 16-bit code is to be used. At the target address Badd(1), various data processing operations 230 are carried out using the 16-bit instruction set. A branch instruction 240 is then performed to return to the 32-bit instruction set. In particular, the branch instruction 240 has a target address Badd(2), referring to a portion of 32-bit code, to which zero is added in order to return the T bit to a zero state. At the target address Badd(2) various data processing operations 250 are performed and the processing ends 260.

When a switch is made between the two instruction sets by changing the T bit in the program counter 130, the actual switch-over by the multiplexer 90 may be delayed to allow for existing instructions currently stored in the pipeline 80.

In summary, the switch between different processing modes (in particular, the use of different instruction sets) can be made by writing a target address and a mode flag (T) to the program counter as part of the execution of a branch instruction.

In an alternative case where the first instruction set is pre-defined and used in existing processors, there may be logical restrictions within the existing first instruction set preventing the normally unused bits of the program counter register 130 from being changed by the instruction set. For backwards compatibility of processors incorporating the second, alternative, instruction set, it may be necessary to employ a short instruction set selection sequence of code to switch in one direction from the first (existing) instruction set to the second instruction set. Since the second instruction set would generally be added at the same time that the switching mechanism is being added, the second instruction set can be defined without the restrictions on accessing normally unused bits of

the program counter register 130. This means that the branching mechanism described above can be used to switch back from the second instruction set to the first instruction set.

An example of an instruction set selection sequence (known as a "veneer") is as follows:

```
Label_Veneer
        XOR           (PC,1)
        Branch        Label
```

In this routine, the current contents of the program counter register 130'' of Figure 3 are exclusive-ORed with 1 to set the T bit to 1. (Alternatively, with the program counter 130' of Figure 2, the current contents could be exclusive-ORed with 10000000000000000000000000000000 to set the T bit).

In an alternative veneer routine, a subtract operation could be used instead of an exclusive-OR operation to change the T-bit of the program counter register 130''. This has the advantage that in some processors, the subtract operation also flushes or clears the instruction pipeline 80.

The following example assumes that the program counter 130'' points 8 bytes beyond the current instruction, and that the current instruction is a 32 bit (4 byte) instruction. Accordingly, to change the least significant bit of the program counter register 130'' to 1, it is necessary to add or subtract the following amounts to the current program counter register contents:

```
        add 1          (to change the T bit to 1)
        subtract 8     (to  compensate  for  the  program  counter
                       pointing ahead of the current instruction)
        add 4          (to compensate for the length of the current
                       instruction)
        ----------
        subtract 3     (total change)
        ----------
```

The instruction sequence used is therefore:

```
Label_Veneer
    SUB         (PC,PC,3)           (replace PC with PC-3)
    Branch      Label
```

In summary, the use of the program counter to store the instruction-set-specifying bit or bits has at least the following advantages:

1.   It provides a single, uniform method of identifying a target routine by representing both the target address and the corresponding instruction set in a single machine word.

2.   The code size is reduced as fewer veneers are required.

3.   The processor performance can be improved as there is no longer a need to execute a veneer on each inter-instruction set routine call.

CLAIMS

1. Data processing apparatus comprising:
    a processor core having means for executing successive program
    instruction words of a predetermined plurality of instruction sets;
    a data memory for storing program instruction words to be
    executed;
    a program counter register for indicating the address of a next
    program instruction word in the data memory;
    means for modifying the contents of the program counter register
    in response to a current program instruction word; and
    control means, responsive to one or more predetermined indicator
    bits of the program counter register, for controlling the processor
    core to execute program instruction words of a current instruction set
    selected from the predetermined plurality of instruction sets and
    specified by the state of the one or more indicator bits of the program
    counter register.

2. Apparatus according to claim 1, comprising:
    a first instruction decoder for decoding program instruction
    words of the first instruction set; and
    a second instruction decoder for decoding program instruction
    words of the second instruction set;
    and in which the control means is operable to control either the
    first instruction decoder or the second instruction decoder to decode
    a current program instruction word.

3. Apparatus according to claim 2, in which:
    program instruction words of the first instruction set are X-bit
    program instruction words; and
    program instruction words of the second instruction set are Y-bit
    program instruction words;
    Y being different to X.

4. Apparatus according to any one of the preceding claims, in which
    the one or more indicator bits of the program counter register are one
    or more most significant bits of the program counter register.

5. Apparatus according to any one of claims 1 to 3, in which the one or more indicator bits of the program counter register are one or more least significant bits of the program counter register.

5

6. Apparatus according to any one of the preceding claims, comprising means for accessing a program instruction word stored in the data memory, the accessing means not being responsive to the one or more indicator bits of the program counter register.

10

7. Data processing apparatus substantially as hereinbefore described with reference to Figures 1 and 2 of the accompanying drawings.

8. Data processing apparatus substantially as hereinbefore described with reference to Figures 1, 3 and 4 of the accompanying drawings.

Patents Act 1977		Application number
Examiner's report to the Comptroller under Section 17 (The Search report)		GB 9411670.4

Relevant Technical Fields	Search Examiner B G WESTERN
(i) UK Cl (Ed.M) G4A (AFP)	
(ii) Int Cl (Ed.5) G06F 9/30	Date of completion of Search 26 JULY 1994
Databases (see below) (i) UK Patent Office collections of GB, EP, WO and US patent specifications. (ii) ONLINE DATABASE: WPI	Documents considered relevant following a search in respect of Claims :- 1-8

Categories of documents

X: Document indicating lack of novelty or of inventive step.

Y: Document indicating lack of inventive step if combined with one or more other documents of the same category.

A: Document indicating technological background and/or state of the art.

P: Document published on or after the declared priority date but before the filing date of the present application.

E: Patent document published on or after, but with priority date earlier than, the filing date of the present application.

&:   Member of the same patent family; corresponding document.

Category	Identity of document and relevant passages	Relevant to claim(s)
A	GB 2016755 A (S E GROVES) See whole document	
X	EP 0324308 A2 (IBM) See whole document	1, 4
A	EP 0306920 A2 (NEC) See whole document	

Databases:The UK Patent Office database comprises classified collections of GB, EP, WO and US patent specifications as outlined periodically in the Official Journal (Patents). The on-line databases considered for search are also listed periodically in the Official Journal (Patents).